# Server端架构简介及技术选型

焦永智

2013/10/28

## 架构简介

### 1.系统职能

提供内网WEB媒质内容管理界面，供产品、运营及编辑人员对产品内容进行编辑及定制、对产品运营数据进行设定及简单监控；依据产品内容定制条件自动抓取数据进行内容更新；自动获取主站CMS频道及分类变更消息并发送邮件通知；管理平台使用者及模块操作权限、记录使用者操作日志；

提供外网HTTP媒质内容访问接口，供前端APP应用获取验证、运营及内容数据，为最终用户实现产品服务及价值。

#### 2.系统拓扑



终端APP产品依赖于移动端业务部CMS系统和移动端业务部外的图片CDN系统、Vlog视频库系统以、WEB主站CMS接口及CMS内容变更activeMQ消息服务。

#### 3.架构说明

移动端CMS系统：

数据库：两台服务器做主从配置，未做读写分离，每日全量备份容灾；

CACHE：目前两台缓存服务器集群配置，可平行扩展；

JOB：两台JOB服务器集群配置，提供一定的高可用性；

CMS后台：单节点部署，内网访问；

REST：目前两台服务器集群配置，前端路由及Ngnix负载均衡，可平行扩展；

外部依赖系统：

图片CDN：提供图片缓存及加速服务，通过http服务到CMS后台服务器回源；

Vlog系统：视频基本信息及视频源，目前通过直接配置数据源进行查库操作，将来计划改为调用Vlog系统提供的内容接口；

视频站CMS：根据频道及分类信息提供视频信息数据；

ActiveMQ：接收视频站CMS频道及分类变更消息，通过邮件服务进行通知。

## 二．代码结构

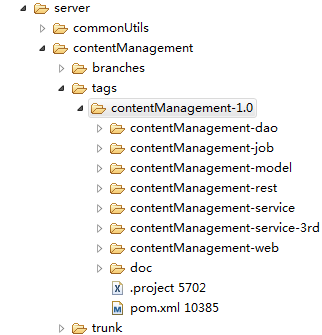
#### 1.代码层级



业务层：细分为两层，CMS本身业务逻辑层和外部依赖业务逻辑层，第三方接口调用抽离出来，与不依赖这些接口的业务部分解耦；

控制层：分为独立两个工程，CMS后台展现部分Controller为典型应用，控制页面逻辑及数据传递；REST服务部分Controller为非典型应用，主要应用SpringMVC对REST风格的实现来发布REST服务。

#### 2.工程目录



contentManagement-1.0 ：父工程，无实质代码，配置父pom文件；

contentManagement-model ：子工程，业务模型工程；

contentManagement-dao ：子工程，持久化层工程，依赖contentManagement-model；

contentManagement-service ：子工程，业务逻辑层工程，依赖contentManagement-dao；

contentManagement-service-3rd：子工程，外部依赖业务逻辑层工程，依赖contentManagement-service；

contentManagement-web ：子工程，CMS管理平台页面展现层工程，依赖contentManagement-service-3rd；

contentManagement-job ：子工程，CMS管理平台内容自动抓取job工程，依赖contentManagement-service-3rd；

contentManagement-rest ：子工程，CMS管理平台内容数据REST服务工程，依赖contentManagement-service；

Model、Dao、Service和Service-3rd为核心代码模块，Web、Job和REST为在核心代码模块基础上构建的具有特定职责的应用模块。

## 三．技术选型

#### 1.选型原则

* 尽量选择门槛低，容易上手的技术，能够快速进入实质开发；
* 尽量选择开发人员用过或相对熟悉的技术，发挥技术积累优势；
* 尽量选择成熟、稳定和资料相对较丰富的技术，规避风险；
* 尽量选择简单、简洁的技术，降低项目复杂性；
* 尽量选择公司要求或者普遍在用的技术，降低外部资源接入成本；
* 适当选择前沿和流行的技术，满足开发人员技术进步的需求；
* 没有完美的技术，只有适合的技术，后期可进行技术更新解决产生的问题；

#### 2.当前选型

* 数据库：MySQL，足够满足预期的数据量级，库表均为InnoDB提供事务支持；
* 连接池：选用功能相对齐全且高并发下相对稳定的C3P0，项目进入平稳期后可以考虑 预研和评估效率明显更高的BoneCP替换的可行性；
* ORM：选用了springJdbcTemplate，组员都使用过，上手快；
* 中间层：Spring框架，bean管理、事务管理等；
* 控制层：选用SpringMVC提供页面控制和REST接口服务；
* 展现层：Html+CSS+Jsp+javaScript+JSTL+JQuery+Ajax+JQueryUI等；
* 数据交换：选用轻量级的JSON，与SpringMVC和JS衔接方便，并且是外部依赖系统统一的数据交换格式；
* JSON转换：JSON LIB功能强大、易用，但性能太低；fastJson低版本性能优异，但随着版本升高性能优势逐渐消失，且BUG相对较多；综合考虑，选用了Jackson，性能稳定且高效；
* 消息中间件：选用ActiveMQ，消息提供方的技术方案；
* 远程调用：选用阿里巴巴的Dubbo框架，服务提供方的技术方案；
* 定时任务：选用比较通用的Quartz；
* 日志记录：选用比较经典的Log4j；
* 缓存框架：选用Memcached对业务数据进行缓存；
* Memcached Client：选用XMemcached，Memcached Client for Java比SpyMemcached更稳定、更早、更广泛，SpyMemcached比Memcached Client for Java更高效，XMemcached 比SpyMemcache并发效果更好，XMemcached更容易与Spring集成等；
* 安全验证：前端公网REST服务接口采用标准的数字信封算法加密解密；内网CMS管理平台使用简单的用户名和密码方式；
* OXM：JAXB2标准的Annotation注释；
* 版本管理：选用SVN，公司统一版本管理工具；
* 工程管理：选用Maven，方便进行jar包统一管理、全局编译打包及子工程级别局部单独编译打包、工程打tag、创建branches和升级开发版本等；
* 其他：apache.httpcomponents httpclient、pinyin4j、javax.mail、commons-fileupload等。

#### 3.问题及改进

* Jar包版本：尽量选择最新released jar包版本，更多的性能及更少的bug，但实际效果需要在使用中进行评估，例如Quartz最新版本在使用中发现它本身版本迭代接口变更产生的异常，最终通过降版本进行规避；
* ORM：目前采用的springJdbcTemplate，但发现springJdbcTemplate很难与cache结合，提供SQL级别的数据缓存，后期考虑替换为Mybatis；
* Memcached：目前使用Memcached的Annotation方式进行业务数据缓存管理，但有局限性，例如不能在一次Service调用中清除多个缓存内容，后期考虑深入学习Annotation方式缓存管理，看看是否有解决方法，或者考虑替换为传统Memcached使用方式进行缓存管理，再或者考虑调整现有Service接口来解决此类问题；
* 其它未暴漏或未发现问题持续观察及解决；
* 新的技术选型，用以解决系统问题和满足新业务需求开发，例如Mybatis引入解决底层cache问题、全文检索引擎Lucene引入用于搜索新业务的需求开发、PUSH技术用于推送新业务的需求开发、阿里云OSS技术用于接入公司新的图片DFS系统等。